Matteo Lavit Nicora
Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing,
National Research Council of Italy,
Via Previati 1/E,
Lecco 23900, Italy
e-mail: matteo.lavit@stiima.cnr.it

Roberto Ambrosetti
Department of Mechanical Engineering,
Politecnico di Milano,
Milano 20156, Italy
e-mail: robertoa@gimac-Int.com

Gloria J. Wiens[1]
Fellow ASME
Department of Mechanical and Aerospace Engineering,
University of Florida,
Gainesville, FL 32611-6250
e-mail: gwiens@ufl.edu

Irene Fassi
Mem. ASME
Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing,
National Research Council of Italy,
Via Alfonso Corti 12,
Milano 20133, Italy
e-mail: irene.fassi@stiima.cnr.it

Human–Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence

To enable safe and effective human–robot collaboration (HRC) in smart manufacturing, seamless integration of sensing, cognition, and prediction into the robot controller is critical for real-time awareness, response, and communication inside a heterogeneous environment (robots, humans, and equipment). The specific research objective is to provide the robot Proactive Adaptive Collaboration Intelligence (PACI) and switching logic within its control architecture in order to give the robot the ability to optimally and dynamically adapt its motions, given a priori knowledge and predefined execution plans for its assigned tasks. The challenge lies in augmenting the robot's decision-making process to have greater situation awareness and to yield smart robot behaviors/reactions when subject to different levels of human–robot interaction, while maintaining safety and production efficiency. Robot reactive behaviors were achieved via cost function-based switching logic activating the best suited high-level controller. The PACI's underlying segmentation and switching logic framework is demonstrated to yield a high degree of modularity and flexibility. The performance of the developed control structure subjected to different levels of human–robot interactions was validated in a simulated environment. Open-loop commands were sent to the physical e.DO robot to demonstrate how the proposed framework would behave in a real application. [DOI: 10.1115/1.4048950]

Keywords: human–robot collaboration, cobot, smart factory, task segmentation, control and automation, robotics and flexible tooling

Introduction

Recent advancements in sensing, computational intelligence, and big data analytics have been rapidly transforming and revolutionizing the manufacturing industry toward robot-rich and digitally connected factories. As reported by IFR [1], the number of collaborative robots (cobots) installed is still very low, with a share of 3.24% of the total market, but with a promising 23% increase of annual installations from 2017 to 2018. Moreover, market research shows that the so-called "mass-customization" of products is already evolving toward a "mass-personalization" [2], raising the need for extremely flexible solutions, such as collaborative robotics. However, effective, efficient, and safe coordination between humans and robots on the factory floor has remained a significant challenge. In order to overcome the current limitations of human–robot collaboration (HRC), an international collaboration composed of U.S. universities (Missouri University of Science and Technology, University of Florida, Case Western Reserve University, and SUNY Stony Brook), the National Research Council of Italy (STIIMA-CNR), and Comau LLC (COnsorzio MAcchine Utensili) has recently launched a novel project called "Intelligent HRC for Smart Factory." The aim is to develop an integrated set of algorithms and robotic testbeds to sense, understand, predict, and control the cooperation of human workers and robots in collaborative manufacturing cells, for significantly improved productivity of hybrid human–robot production systems toward deployment in future "smart factories." The authors of this paper are participants in the above international collaboration, tasked with addressing the integration of the cognition and prediction with the robot's planning and control. The long-term goal of the authors is to develop a multi-layer and modular control structure that allows stable mode switching for flexibility in defining the "optimized" real-time robot response, to safely adapt to human worker planned and unplanned interactions, and to maintain production efficiency. The core of this control structure, labeled as Proactive Adaptive Collaboration Intelligence (PACI), is in charge of modifying the robot motion on the basis of inputs related to the predicted human actions and body motion trajectories and to the preplanned robot trajectories and task breakpoints. Kinematic segmentation is proposed for effectively parsing the data received from the predictor and for identifying impact on current/preplanned robot trajectories. The specific task at hand is then kinematically adapted in real-time for safe controlled robot motions, to optimize the collaboration and to mitigate production disruptions. The authors, building upon the proof of concept provided in Ref. [3], envision a robotic system having the following features: *flexibility* (seamless adaptability of the system to a wide variety of applications and advanced customizability of the product), *accessibility* (intuitive, fast, and easy programmability of the robotic system, accessible to non-expert users), *modularity* (enhanced reusability of the code that allows developers to easily update the system), *safety* (real-time awareness and response capabilities ensuring safety of humans, robots, and equipment co-operating within the manufacturing cell), and *productivity* (smart robot reactions aimed at mitigating productivity disruption). In order to develop a robotic system able to provide a framework coherent with the long-term goal of the project and capable of providing the above desired features, the authors propose a control architecture composed of two multi-layer modules, as shown in Fig. 1. The first component, the Offline Module, is equipped with a graphical user interface (GUI) that takes as input the requests of the user and feeds the processed information to a second module, which is in charge of the kinematic segmentation of the task(s) and the preplanning and management of the created segments. The second component, the Online/Real-time Module, receives as input the information provided by the first module, and collected data about the human presence and the

[1]Corresponding author.

Manuscript received May 22, 2020; final manuscript received June 6, 2020; published online December 16, 2020. Assoc. Editor: Y. Lawrence Yao.

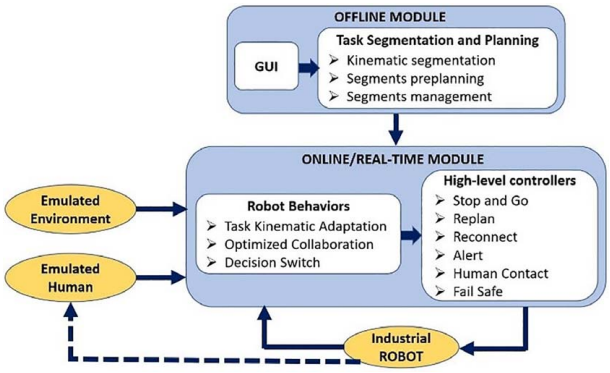

Fig. 1 Control structure

external environment in order to perform the online kinematic adaptation of the robot motion and achieve optimized collaboration with the operator. A decision switch layer is triggered to activate in real-time the high-level controller that best suits the specific human–robot interaction scenario at hand. Due to the early stages of the project at the time of the authors' research, the control architecture has been implemented within an emulated environment with open-loop commands sent to the physical e.DO robot in order to demonstrate the potentiality of the system in a real collaborative application.

Section 2 provides a brief presentation of the materials and methods exploited for this research. Section 3 provides the reader with a general overview of the state-of-the-art of human–robot collaboration. Detailed descriptions of the Offline Module and the Online/Real-Time Module are reported in Secs. 4 and 5, respectively. A case study is then analyzed in Sec. 6 in order to evaluate and validate the capabilities of the developed robotic system. Finally, a discussion about the significant results of the research is presented in Sec. 7 followed by a conclusive summary of the work in Sec. 8.

Materials and Methods

The authors chose to exploit ROS Melodic Morenia [4] running on an Ubuntu 18.04 Bionic Beaver system. It is today's widely adopted standard platform for robotic research and allows the development of a scalable system, easily adaptable to several robotic cells, guaranteeing high levels of maintainability of the system itself. The availability of numerous path planning algorithms and robot controllers in the open-source library MoveIt! [5] was also leveraged in order to develop the envisioned control architecture. All the codes have been written in C++. The work was validated within an emulated environment with open-loop commands sent to an e.DO robot by Comau [6], demonstrating an experimental collaborative assembly scenario with simple building blocks.

State of the Art

In order to achieve seamless human–robot interaction, many solutions have been proposed in the past, but success is limited due to the disruption of system productivity, caused while ensuring the required level of safety. A very interesting approach has been proposed in Ref. [7] in which the authors envision "complex robot behaviors to emerge in real time from the interplay of several concurrently running elemental controllers." A predefined library of skills optimized beforehand is made available by the authors of the mentioned paper, and their proper reactive sequencing is obtained through so-called Behavior Trees, introduced in Ref. [8]. Additional collaborative approaches are available in the literature. For example, a deformation-tracking impedance control method was validated with a robot performing a cooperative assembly task with the human worker acting as the time-varying environment [9]. Using a parallel combination of a baseline time-invariant controller and a safety controller enforcing a time-varying safety constraint, others are working to establish a set of design principles for a safe and efficient robot collaboration system (SERoCS). Their approach consists of efficiency and safety goals treated separately, modularized structure, compatible with existing robot motion control algorithms, online safety controller, robot motion confined to safe regions according to predicted human motion, and reduced computations by modeling humans as single or multiple spherocylinders with portions of the control policy solved offline [10]. A collision avoidance strategy is presented in Ref. [11] for online replanning of the robot motion and creates a safe network for unsafe devices (distributed layers of data cross-checking and validation of sensors, PLCs, PCs) as an infrastructure for achieving functional safety. An optimal control problem formulated for a physical HRC-based robot motion control is augmented with a social HRC in Ref. [12], improving interactions in assembly tasks by increasing the human worker's trust in his/her robot partner. However, because the generation of safe robot trajectories is limited due to the inherent uncertainty of robot trajectory execution time, STIIMA-CNR estimated a confidence interval on robot trajectory execution time for scenarios in which human–robot space sharing is required [13]. Another relevant research topic for the field of collaborative robotics is the way cobots are managed and taught. It is of fundamental importance to provide human operators with intuitive interfaces that ease the process of communication and therefore leave the operator free to concentrate on the task and goals at hand. Many solutions are available in the literature and are currently being investigated, such as *walk-through programming*, in which the operator physically moves the end-effector of the robot through the main positions of the task [14] and *programming by demonstration*, in which the robot is not purely reproducing the motion of the operator but is also able to generalize it into new scenarios [15]. *Offline programming*, instead, aims at the minimization of the downtime of the robot by using software tools to virtually replicate the shop floor on a computer [16]. Finally, much interest has recently been devoted to augmented and virtual reality for manufacturing applications. Novel examples of their application have shown how these approaches can yield improved productivity of the system and enhanced human safety [17]. It is clear that many solutions have been proposed to address specific applicative cases but, in today's rapidly changing industrial environments, each product may have particular requirements. In this paper, a flexible and modular framework is developed that accommodates different HRC approaches and exploits them whenever needed. Thus, providing an efficient integration of multiple controller logic via task segmentation enforcing behavioral preferences/constraints.

Offline Module: Task Segmentation and Planning

A general observation of real industrial cases leads to the realization that every scenario can be thought of as composed of a series of discrete subtasks (set of actions). These subtasks can represent a movement of the robot between two configurations in space, a specific action of the end-effector performed by the robot in a certain position, or a combination of the two. The offline module in Fig. 1 has been developed in accordance with this logic: a collaborative application is divided into subtasks in a process called "segmentation" and each subtask is addressed to as a "segment." This approach opens up the possibility to independently manage each segment both in terms of how the robot's motion trajectory is planned offline and of how the robot will react online to unexpected obstacles (e.g., human encounters) during the execution of the mentioned trajectory. The authors expect the segmentation process to yield flexibility, easy programmability, and customizability in the definition of the production process. A dedicated algorithm, representing a node in the ROS environment, has been written in order to implement said approach and is composed of four sections: *input reading* (translation of the information given by the user through the GUI into efficient code language), *offline planning* (trajectory

generation for each segment according to the user's requests), *segments connection* (segments analysis to ensure proper connection between sections), and *segments execution* (to ensure correct timing and sequencing for the execution of the segments).

Graphical User Interface. In order to enhance the accessibility of the robotic system and lower the skill threshold required to program the robot, an intuitive GUI has been developed. Thanks to this tool, the segmentation is performed in an iterative offline process by the user being guided through the full definition of robot's task and prompted for specifying the levels of human–robot interaction allowed for each segment. This anticipatory offline preplanning approach embeds into the robot controller the underlying intelligence for enabling planned and unplanned industrial collaborative scenarios to occur seamlessly in real-time. Each time a new segment is created, a choice of offline planning techniques and the type of permissible online robot behavior(s) are assigned to it by the user. According to this selection, more detailed information is requested in order to completely define the characteristics of the segment that are later used in generating the robot's reaction/interaction to the presence of a human within its workspace. Once the robot's task has been programmed segment by segment, the algorithm takes care of all the remaining offline planning steps (input reading, offline planning, and connection analysis). As said, the first step performed by the ROS node is a process of input reading. All the information provided by the user through the GUI is read by the algorithm and translated into a code-efficient language, exploiting structure objects of C++ to store the data related to each created segment.

Offline Planning. With the aim of ensuring easy programmability while maintaining great freedom in the definition of the task, a set of offline planning techniques is provided to the user for selecting the technique best suited for the task. In order to address the main needs of common industrial applications, four methodologies are made available and collected in Table 1.

The first offline planning technique, called *User-defined Algorithm*, addresses all the cases in which the operator is only interested in defining start and goal configurations of the robot for the segment, not the particular trajectory connecting them, and has no a priori knowledge of the task. First, the user chooses to specify the robot configuration in joint space or Cartesian space. According to this choice, either the pose of the end-effector or all joint angles are required in order to determine the start and goal configurations of the robot for the segment. A list of planners, offered by the OMPL library in MoveIt!, can then be used to select the best suited algorithm. The code developed by the authors takes as input these requests and gives as output a feasible trajectory that respects the hardware limits set for the robot. Two scaling factors are also available if, for any reason, the user needs to further reduce the preset speed and acceleration limits.

Table 1 Summary of the implemented skills

Implemented skills	
Offline planning techniques	User-defined algorithm Human occupancy volumes Relevant trajectory Tool operation
High-level controllers	Stop and go Replan Reconnect Alert Allowed contact Fail safe
Robot behaviors	Limited time stop Unlimited time stop Robot trajectory (non-restrictive) Robot trajectory (restrictive) Human contact

The second approach, called *Human Occupancy Volumes* (HOVs), is inspired by Pellegrinelli et al. [13] and has been developed in order to address the case in which the user is still interested in simply defining start and goal configurations of the robot for the segment, but experimental data about the human presence for the specific task are available. In Ref. [13], a cooperative operation is described as a spatial and statistical distribution of HOVs. Through an experimental campaign, each point in space is assigned a probability of being occupied by the operator during the specific task and points belonging to a certain probability range are grouped into an HOV. It is immediately clear that a trajectory avoiding all the volumes has a higher execution time because a longer path is needed to circumnavigate the whole workspace, but a smaller variability since, ideally, the probability of encountering the operator is bound to zero. On the other hand, a trajectory crossing one or more of the HOVs is shorter and therefore faster but, at the same time, more likely to undergo variations due to stops or changes in speed required to avoid collisions with the operator. The procedure above is used in this research in order to define a priori the path to be followed by the manipulator. The computed path is, statistically, the one expected to minimize the execution time by simultaneously considering its length and the probability of disturbance. For the sake of brevity, the implemented procedure is not explained in detail here and, therefore, the authors suggest referring directly to Ref. [13] for further information. The role of the user in all of this is simply to input start and goal configurations of the robot for the segment either in joint or Cartesian space, upload the previously defined volumes and, if needed, specify scaling factors for the limits on speed and acceleration.

As opposed to the previous cases, there are many situations in which the user has to completely define the trajectory of a segment. For instance, if the robot has to perform some kind of activity along with its motion (e.g., arc welding, cutting, and pouring molten metal), a particular path, defined point by point, must be followed. The *Relevant Trajectory* offline planning technique has been developed specifically for this purpose. The path is built by the user point by point, either in joint or Cartesian space. Once all the points have been defined, the algorithm automatically analyzes them to create a feasible trajectory that respects all the limits of speed and acceleration. The offline planning allows the user to lower the speed and acceleration limits via specification of a zero-to-one scaling factor.

The last mode implemented by the authors for the offline planning of the task is called *Tool Operation*. The aim of this mode is not to plan a movement of the robotic arm between two configurations but to plan an action performed by the end-effector in a certain pose. In this mode, the user specifies the position inside the workpiece where one of the actions represented in Fig. 2 has to be performed: *Open* (open the gripper's prongs to a certain distance), *Close* (close the gripper's prongs to a certain distance), or *Wait*. The "Wait" operation enables direct contact with the manipulator, for example in order to inspect the end-effector or perform an operation on the carried workpiece. Since this operation is performed while the robot is stationary, no harm can be done to the operator. The motion of the manipulator is allowed to restart only when the operator gives a confirmation signal by pressing the ENTER key on the keyboard.

As already introduced, one of the goals of the authors is to guarantee the modularity of the robotic system. For this reason, this ROS node has been structured so that any new offline planning technique can be simply added to the list of available choices by introducing the dedicated algorithm inside the code.

Segment Connection and Execution. Supposing that the user has defined all the segments that compose the task at hand and that the robot's trajectories have been computed offline by the

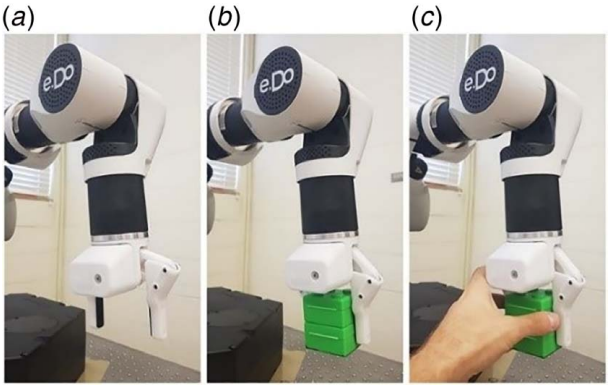

Fig. 2 (*a*) Open gripper, (*b*) close gripper, and (c) wait operation

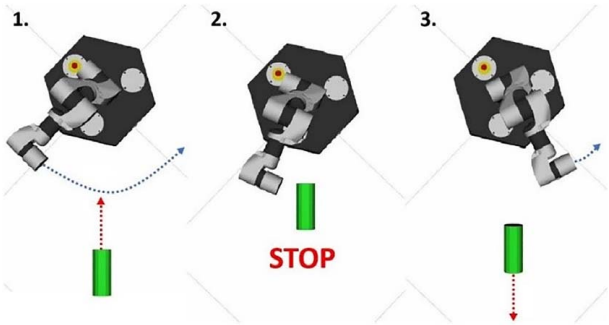

Fig. 3 Representation of the functionality of the Stop and Go controller

algorithm, an additional step is required in order to ensure the correct connection of the subtask trajectories/actions. Two sequential segments can be considered properly connected if the final waypoint of the first one coincides, within a certain tolerance, with the starting waypoint of the following one. The algorithm therefore performs a complete analysis of the generated paths and automatically creates connections where needed, in order to compensate for any mistake or inaccuracy of the user. The sequencing of the connecting segment is critical during this step. For instance, considering a sequence <*Segment_1, Segment_2*>, if a connection is needed between the two segments, the resulting sequence must be <*Segment_1, Connection_Segment, Segment_2*>. The last duty assigned to this ROS node is to make sure that the execution of each segment is commanded with the correct timing. In particular, two conditions must be satisfied before the algorithm is allowed to start the execution of a segment: the readiness of the particular high-level controller responsible for the supervision of the subtask must be ensured and a feedback signal communicating the successful completion of the previous segment must be received.

Online Module: Behaviors and High-Level Controllers

The Online/Real-Time Module in Fig. 1 provides the robotic system with a set of capabilities that can be used to react to different changes in the manufacturing environment. In particular, the concept of "robot behavior" is introduced as a reactive switching logic that enables the manipulator to activate in real-time the high-level controller best suited for the specific human–robot interaction scenario. Due to the segmentation process, the user is able to assign a choice of behavior independently to each segment. In order to implement this second module, two codes, both nodes inside the ROS environment, have been developed. The first one provides a set of high-level controllers, while the second one is used to implement a series of cost function-based switching logics (behaviors).

High-Level Controllers. A high-level controller represents a particular approach available to the robot for its interaction with the environment and the human operator. Having a set of possibilities enables the robotic system to adapt to a great variety of situations typical of an industrial environment. For this reason, the authors implemented six different high-level controllers, grouped in Table 1, within a modular structure that allows for additions to the robot's portfolio of high-level controllers to be easily made.

The first controller developed for the project is called *Stop and Go*, and a representation of its functionality is depicted in Fig. 3. The distance between the body of the robot and any obstacle sensed in the environment during the execution of a trajectory is constantly monitored. If the distance decreases to a certain predefined threshold, an immediate stop of the robot motion is commanded in order to avoid a possible collision. Similar, to safety-rated monitored stop (SMS) approaches found in the literature, the robotic system remains active after the stop so that, as soon as the obstacle moves far enough away from the robot, the motion can be restarted in order to complete the original trajectory. The threshold is set at code level and represents minimum distance between the moving robot and the environment/operator considered acceptable in terms of safety.

A different obstacle avoidance approach has been implemented for the second controller called *Replan*. As before, the robot is stopped if its distance from an external obstacle drops to a certain predefined threshold. At this point, instead of waiting for the original path to be cleared, the robot looks for a new feasible trajectory from its stop position to the goal of the segment and immediately executes it. This instantaneously static replanning operation is performed considering a virtual obstacle of increased dimensions so that the new path is forced to maintain a minimum value of clearance from the real physical object and no unsafe maneuver is allowed. If, for any reason, the robot should move inside the virtual volume of the inflated obstacle, this would be considered a collision in the emulated environment and the robot's motion would be immediately stopped, while in the real world, a safety clearance would still be maintained. A representation of this procedure is shown in Fig. 4.

A variant of this solution is proposed for the third high-level controller, called *Reconnect*. Again, the robot stops in front of the obstacle and looks for a new path but, instead of replanning all the way to the goal of the segment, it tries to reconnect to the original trajectory. Basically, the algorithm analyzes the points of the remaining part of the original trajectory to find the first one not obstructed by the obstacle. A path connecting the stop position of the robot and that point is then computed and augmented with the section of the original trajectory that brings the robot to the goal of the segment.

A different idea has driven the development of the *Alert* controller. The above controllers are aimed at giving priority to the external environment and adapting the motion of the robot to it. The goal of the Alert controller is to provide the robot with the ability to communicate its need for a clear path. In practice, this has been realized

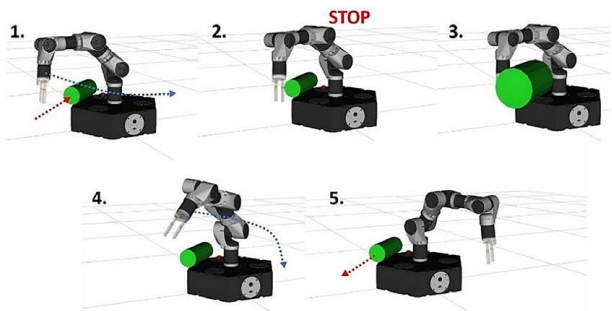

Fig. 4 Representation of the functionality of the Replan controller

by emitting a sound alarm every time that an obstacle is detected in the proximity of the manipulator.

The *Allowed Contact* controller is another option, which is automatically activated every time that the user creates a "Tool Operation" segment using the "Wait" action. Since the robot is stationary in a certain position and no harm can be done to the operator, the distance threshold for external obstacles is deactivated in order to allow direct contact with the manipulator. The safety checks are immediately restored as soon as the operator presses the ENTER key (confirmation signal) and the robot starts moving again.

Finally, a *Fail-Safe* controller has been developed so that, every time that an emergency situation occurs, the system is shut down and maximum priority is given to the safety of the operator. The effect is similar to what an emergency-stop button would do and, in order to restart the operation, the whole system must be rebooted.

Robot Behaviors. The second major component of the Online/Real-Time Module in Fig. 1 is the determination of an optimized human–robot collaboration and subsequent decision switch logic in charge of activating the best suited high-level controller. In order to guarantee flexibility and modularity of the robotic system, a series of "Robot Behaviors" have been implemented in the form of cost function-based switching logics. Every behavior is associated to a specific group of high-level controllers and, by means of a dedicated cost function, evaluates a "cost of activation" for each one of them. The cost represents the impact of the activation of a specific controller on the performance of the system in terms of both safety and productivity. The high-level controller with the lowest cost gets activated in real-time by the switching logic, with the exception of the occurrence of any emergency situation in which case the Fail-Safe controller immediately overrides every other command. Each cost function is structured as the sum of three components: *base cost* (constant value used to give preference to a specific controller when the other parameters are irrelevant), *distance cost* (accounts for human safety, inversely proportional to the distance between manipulator and the closest obstacle (e.g., human body part)), and *delay cost* (account for productivity, proportional to the duration of any stop of the robot motion caused by an obstacle)

$$\text{Cost} = C_{base} + (C_{dist} \div \text{Distance}) + (C_{delay} \times \text{Delay}) \quad (1)$$

Each decision switch logic is equipped with a table containing the values of the three coefficients in Eq. (1) for each active controller. The values set in this table, and the specific controllers considered are what differentiates one behavior from the other. A representation of the online procedure implemented with this ROS node can be seen in Fig. 5.

As the execution of a segment starts, the behavior selected by the user is immediately activated. The cost of each active controller is evaluated on the basis of data collected from the external environment (obstacle distance and induced delay), and the least expensive controller is activated in real-time. The result of such an approach is somewhat similar to what is suggested in Ref. [7]: complex robot behaviors emerge in real-time from the interplay of properly sequenced high-level controllers. Further noting, due to the modularity of the system as many behaviors as needed can be implemented. For the research presented herein, the following are the behaviors developed and implemented to represent different scenarios one anticipates for human–robot interaction within a smart factory:

- *Limited Time Stop*: exploiting the Stop and Go and Replan/Reconnect controllers, this behavior allows the manipulator to stop in front of an obstacle and decide whether to wait for the path to be cleared or to replan around it;
- *Unlimited Time Stop*: version of the previous behavior that forces the robot to wait for the path to be cleared, no matter for how long;
- *Robot Trajectory (Non-Restrictive)*: exploiting the Alert and Stop and Go controllers, the robot is able to alert the operator if an obstacle is disturbing a critical operation and to stop in case a collision is imminent;
- *Robot Trajectory (Restrictive)*: version of the previous behavior that does not leave any possibility for modification of the robot's motion. If an emergency situation occurs, the system is shut down by the Fail-Safe controller;
- *Human Contact*: this behavior is simply used to activate the Allowed Contact controller in order to allow direct interaction with the manipulator until a confirmation signal is given by the user.

Case Study

In order to validate the control architecture of Fig. 1 incorporating both the Offline Module and Online Module operating per the online switching logic depicted by Fig. 5, an experimental activity has been performed within a virtual environment, emulating dynamic data related to the movement of the operator's forearm and considering the rest of the body as a static obstacle. A random component of ± 1 cm is introduced as a fictitious sensor tolerance. Refresh rate has been set to 5 Hz, and no delay is assumed. Open-loop commands are also sent to a physical e.DO robot, equipped with a mechanical two-prong gripper able to perform pick and place operations, in order to demonstrate the potential of

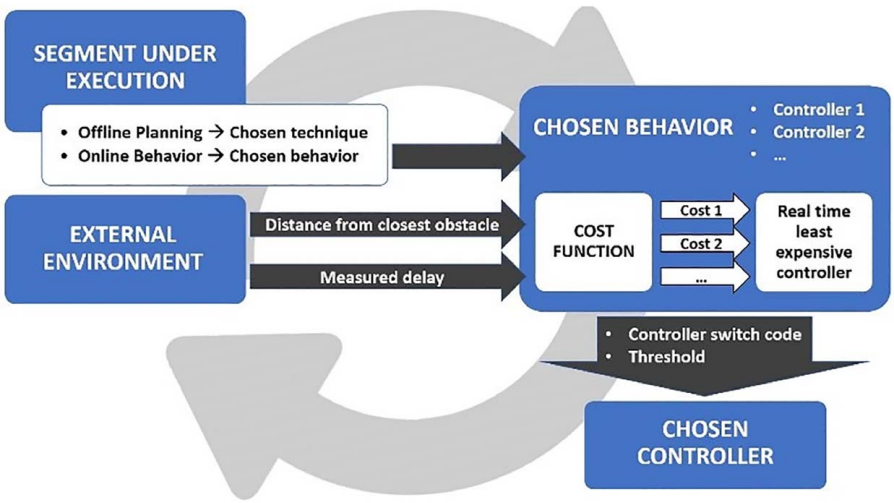

Fig. 5 Working principle of the online switching logic

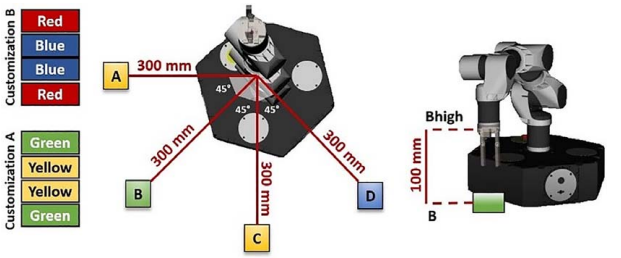

Fig. 6 Schematic of the two customizations of the product and testbed setup

the control architecture in a real application. The authors chose to validate the robotic system by means of a simple assembly task, representative of common industrial scenarios. The manipulator and human share the same workspace, and the assembly of the product is carried out in a collaborative manner. As depicted in Fig. 6, the product under analysis is made up of four components, in this case colored building blocks (as labeled), which are placed in different locations in the workspace [components inserted into the assembly process: Location A (process starting point) – Yellow or Blue Blocks; Location B – Green Blocks; Location C – Yellow Blocks; Location D – Blue Blocks]. Some of the assembly operations are assigned to the human (green/red blocks), others are assigned to the robot (yellow/blue blocks), and some direct interaction between the human and robot is required. During this collaboration, the human may intersect the motion of the manipulator, generating the need for smart robot reactions in order to ensure the safety of the operator while at the same time attempting to minimize the disruption of production efficiency.

Always with reference to Fig. 6, two customizations, each consisting of an ordered stack of blocks assembled from a set of components (herein, the four color choices of blocks), are considered and used to test the flexibility of the system. The difference between the two versions is the location of their components. Either the robotic task must be reprogrammed, or the operator can easily adapt to the new duties. As shown in Fig. 7 (images #1–8), the robot is assembling Customization A. The robot moves from a vertical stand-by position toward the first component in location A (#1). The part is moved toward the second component, in location C, while the operator prepares the assembly area located in B (#2). Once the robot has completed the assembly of the first two parts (#3), it moves away and waits for a confirmation signal given by the human that has to inspect the workpiece (#4). After the signal, the robot picks up the subassembly and starts moving it toward the common assembly area where the operator places the third component with the correct orientation (#5). Before further assembling, the robot once again lets the operator inspect the carried workpiece to ensure the alignment of all the parts (#6). When allowed, the robot performs its last assembly task (#7) and moves back to its stand-by vertical position, while the operator assembles the fourth and last component and retrieves the finished product (#8). In order to program the robotic collaborative task at hand, the developed GUI was used to independently characterize each segment (SEGM). Regarding the offline planning technique, the segments requiring a motion of the robotic arm use the User-Defined Algorithm mode, while the segments representing an action of the end-effector exploit the Tool Operation mode. In terms of online robot behaviors, the authors made the following choices:

- Large motions of the manipulator use the "Limited Time Stop behavior" in order to smartly react to possible interactions with the operator (segments 1, 5, 18, and 19);
- Operations that require high precision and must not be disturbed, such as insertions or gripping/releasing of components, exploit the "Robot Trajectory (Restrictive)" behavior (segments 2, 3, 6, 7, 10, 11, 15, and 16);

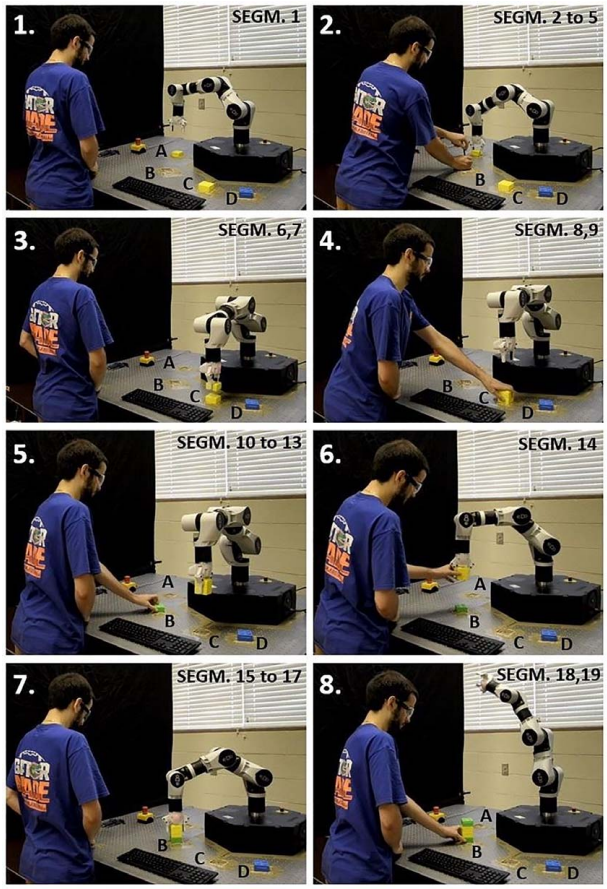

Fig. 7 Main assembly steps: customization A

- Operations that are still critical but with a lower level of required precision are assigned the "Robot Trajectory (Non-Restrictive)" behavior (segments 4, 8, 12, and 17);
- Segments used for inspection of the workpiece and of the subassembly use the "Human Contact" behavior (segments 9 and 14); and
- Segment 13 exploits the "Unlimited Time Stop" behavior because, if interference with the human happens, it will most likely consist in an obstruction of the goal of the segment and therefore no replanning is possible.

Moreover, three levels of human–robot interaction for each customization of the product have been analyzed. The *first case* represents a situation of perfect synchronization, and the motion of the robot is never disturbed by the presence of the operator. For the *second case*, the fixturing of the common assembly area takes slightly longer than expected, meaning that the robot finds its path obstructed by the operator when moving the first component from location A to location C and waits until the operator moves away and the path is clear (segment 5, limited time stop). The *third situation* analyzed addresses the case in which an unexpected issue occurs in the fixturing process, forcing the operator to keep working in the area for an even longer time. Therefore, once again the robot finds its preplanned path (segment 5) occupied by an obstacle and has to react in order to minimize the disruption of productivity. This time it replans its path and goes around the operator.

Results and Discussion

Offline performance: As expected, the graphical user interface sensibly enhanced the accessibility of the system, enabling users to intuitively program the robotic task with minimal advanced

knowledge. Moreover, timers have been implemented in the codes in order to measure the amount of time required for all of the offline operations. Regarding the scenario under analysis, the measured time is equal to around 780 ms. On the other hand, the time needed for the manual data insertion by the user was not precisely measurable, but in the order of minutes. Given the fact that more complex scenarios would require longer times both for the computational part and for the manual data input, it is safe to say that the latter is the process with the highest impact on the offline performance of the system. Using the GUI, this step is required only the first time that a process is introduced in the production system. The interface enables the user to store the program, creating a library of scenarios that are always accessible for immediate use according to production requirements. Moreover, the ability to reopen and modify already existing programs significantly eased the process of customization of the product. Instead of reprogramming the whole robotic task for Customization B from scratch, the user was able to open the program of Customization A, modify it according to the new process, and store it as a new independent scenario, therefore saving much time in manual data input. For all these reasons, the system is demonstrated to have great flexibility, adaptability, and customizability in its offline capabilities, fundamental features for Pull Manufacturing and Industry 4.0 environments.

Online performance: The plot in Fig. 8 reports the accumulation of execution times measured for the three levels of interaction applied to the scenario of Customization A. As shown, the three lines coincide up to the end of segment 4 (robot trajectory, non-restrictive) since, up to that point, no human interaction occurs for any of the cases and therefore the same execution times are measured. Segment 5 is where the three cases start to differentiate. Case 1 represents an efficiency reference, being a situation of perfect synchronization. Cases 2 and 3, instead, generated longer execution times due to the disturbance provoked by the operator's forearm along the path of the manipulator. After that, since no further delay is induced, the three curves develop in parallel up to segment 13, where, once again, longer times where measured for Cases 2 and 3 due to the human presence.

In order to evaluate the online performance of the robotic system, it is interesting to analyze how the robot reacted to the emulated human's interference that occurred in segment 5. Figure 9 presents the trend of the cost of activation of the controllers during the execution of segment 5 for Cases 2 and 3.

In Case 2, the cost of the Stop and Go controller remains the lowest for the entire duration of the motion. Consequently, the reaction of the robot was to stop in front of the obstacle, wait for the path to be cleared and then restart its motion along the original trajectory. For Case 1, 4.5 s was needed to complete the segment 5. For Case 2, the authors measured a time of completion of 7.3 s, accounting for the time in which the robot was paused due to the human's presence. If, instead, the robot would have chosen to immediately replan around the forearm, the goal of the segment would have been

Fig. 8 Plot of the execution times for the three interaction cases of customization A

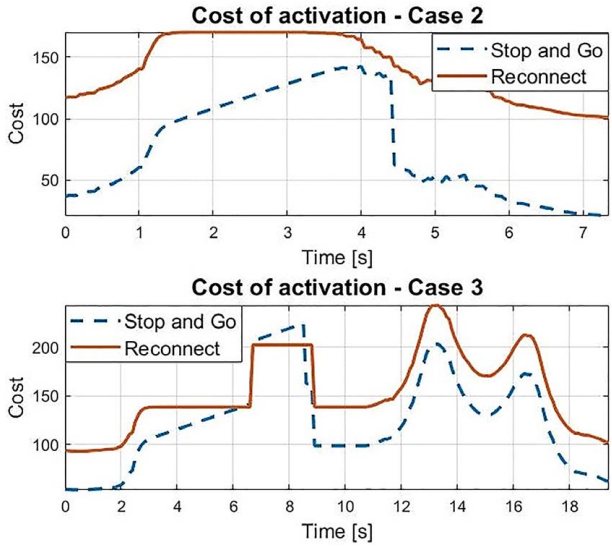

Fig. 9 Trend of the costs of the controllers for the execution of segment 5

reached in 14.7 s. Therefore, the Stop and Go reaction actuated by the robot is the most time saving one for Case 2. On the other hand, regarding Case 3, the cost of the Stop and Go controller becomes larger than the cost of the Reconnect controller at 6.8 s, meaning that a switch is triggered. In practice, the robot stopped in front of the obstacle. But, since the induced delay was excessive, the system decided to replan its path and complete the segment. The execution time measured for this reaction was 19.1 s, versus the 21.5 s that would have been needed if the robot had to wait for the operator to clear the path. This means that, once again, the reaction chosen by the robot was the most appropriate for the situation under analysis, with a consequent 11% reduction of the productivity disruption.

Parameters analysis: To quantify the performance of the HRC control architecture, a series of parameters on safety and productivity of the robotic system are now defined. Figure 10 provides a representation of the parameters under analysis: a proximity threshold (t), used to stop the robot within a predefined distance from the

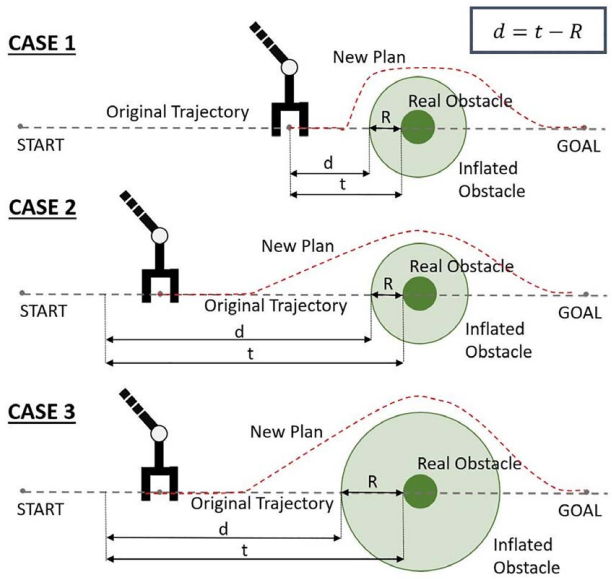

Fig. 10 Schematic representation of reaction distance and inflation radius

obstacle, and a virtual inflation radius (R), used to guarantee a certain clearance from the obstacles in the execution of replanned trajectories. The first one (t) can be considered a sort of "reaction distance," the second one (R) an indicator of safety, while the execution time represents the productivity of the system. Figure 10 presents three examples, each one of them characterized by different values for these parameters. Due to the different shapes of the replanned trajectories for the three cases, the authors expect sensible impacts on the execution times.

Noting that the amount of inflation must be limited according to the position in which the robot is stopped, an inflated obstacle overlapping with the body of the manipulator would represent a "virtual collision" that must be avoided. For this reason, a third parameter (d) is defined in Fig. 10. By making sure that its value is greater than zero, the limit is respected. The experimental campaign consists in repeatedly executing the trajectory of segment 5 with different values of t and R, measuring for each execution the time needed by the robot to move from the beginning (start) to the end (goal) of the segment. The analysis has been performed by iteratively fixing a certain value of d while varying the value of virtual inflation R (and consequently the reaction distance t). For each fixed d, nine values of R are considered, and the obtained points have been interpolated to obtain the curves represented in Fig. 11. Moreover, for each point (a set of d and R), 20 executions of the trajectory have been performed in order to average the intrinsic variability of the exploited planner. Analyzing each curve of the plot in Fig. 11, the trend is to have an increasing execution time as the value of inflation grows. In fact, comparing Cases 2 and 3 in Fig. 10, a bigger inflation leads to a longer replanned trajectory and therefore longer execution times. Considering now the five plotted lines and a fixed value of inflation, it is clear that as d gets smaller, the average execution time rises. Always with reference to Fig. 10, this behavior can be explained by the fact that a short reaction distance (t) yields a replanned trajectory characterized by "sharp turns" (Case 1), while a long reaction distance leads to a smoother path (Case 2). Since the manipulator has limits on the maximum acceleration that can be produced by its motors, sharp turns sensibly slow down the execution of the trajectory, while a smooth path can be executed much faster.

Summarizing these results, it is clear that, in order to have lower execution times and therefore maximize the productivity of the system, a large reaction distance (t) and a small virtual inflation would be best, in general. On the other hand, the inflation radius (R) is an indicator of safety and its reduction would generate higher risks and discomfort for the operator. Also, having a large reaction distance (t) would cause much more disturbances to the robot motion, as it would try to adapt its trajectory according to obstacles that are still far from its body. As seen in Fig. 11, a nonlinear relationship is also observed. That is, the average execution time (inverse indicator of productivity) is not necessarily monotonically increasing with an increase in R, indicator of safety. Consequently, further modeling is needed to arrive at an optimal set of these parameters able to simultaneously maximize operational safety and productivity. The authors believe that the results obtained with this experimental campaign could represent useful information for the future developments of the robotic system. In fact, knowing the impact that the robot's reaction distance has on the performance of the system could be a powerful driver in the definition of an optimal value for the "time span" to be covered by the predicted data envisioned for the long-term project.

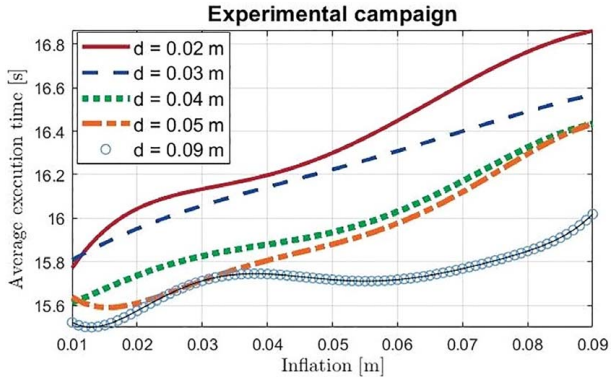

Fig. 11 Execution time curves as a function of reaction distance and inflation radius

Conclusion

In this paper, robot reactive behavior intelligence was achieved via cost function-based switching logic activating best suited high-level controllers. A series of offline planning techniques, high-level controllers, and robot behaviors have been made available for the independent characterization of the segments. The Task Segmentation was shown to provide flexibility, adaptability, and customizability to the robotic system, in line with the Industry 4.0 requirements. The resulting PACI and switching logic was validated to be effective in guaranteeing safety of the operator while reducing negative impacts on the productivity by dynamically adapting the robot's motions given predefined execution plans for its assigned tasks and detection of emulated human actions. Further analysis was conducted to extrapolate the influence of two parameters (*reaction dist*ance and *inflation radius*) on operational safety and productivity of the system. These preliminary findings will serve as a powerful driver for the definition of the time span to be covered by the predicted human motion data envisioned for future developments of the research. Future work will involve investigation of potential stability issues that may arise with the switching logic due to uncertainties in predicting human motion, introducing real-life sensing of multiple dynamic obstacles, testing for "sensor failure" scenarios, and expanding the cost functions to consider a greater set of data (e.g., safety indicators and predicted human motion parameters).

Acknowledgment

The authors wish to acknowledge Jack Wittmayer, undergraduate researcher at University of Florida, for his development of the GUI in support of the research. This research was conducted under the University of Florida J-1 program and in collaboration with the Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing – National Research Council (STIIMA-CNR) of Italy.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request. The authors attest that all data for this study are included in the paper.

Funding Data

- Funding was provided by Politecnico di Milano scholarships "Thesis Abroad – Borsa di Studio Tesi all'Estero" for both Matteo Lavit Nicora and Roberto Ambrosetti and by the National Science Foundation/National Robotics Initiative (NSF/NRI) 2.0 award entitled NSF/NRI: INT: COLLAB: Manufacturing USA: Intelligent Human–Robot Collaboration for Smart Factory (Award I.D. #:1830383). Any opinions,

findings, and conclusions or recommendations expressed are those of the researchers and do not necessarily reflect the views of the National Science Foundation.

Nomenclature

- d = distance between the position of the robot and the inflated obstacle, must be greater than zero to avoid a "virtual" collision
- t = reaction distance, distance threshold within which the robot will react to the presence of an obstacle inside the robot workspace
- R = virtual inflation radius of an obstacle, minimum distance acceptable for a robot trajectory to pass close to an obstacle
- C_{base} = base cost of robot behavior, constant value specified to give a preference
- C_{delay} = delay cost coefficient, constant
- C_{dist} = distance cost coefficient, constant
- Cost = cost of activation of robot behavior
- Delay = time delay induced by any stop of the robot motion due to the presence of an obstacle, measured in terms of number of code cycles spent, while the robot is stopped
- Distance = distance between manipulator and the closest obstacle
- ROS = robot operating system, robotics middleware (collection of software frameworks for robot software development; libraries and tools)
- SEGM # = robot application divided into consecutive segments (robot movement and/or end-effector action), SEGM # designates segment identified by #

References

[1] International Federation of Robotics, 2019, "IFR Press Release," IFR Press Conference, Shanghai, China, Sept. 18, https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd, Accessed October 18, 2019.
[2] Fenech, Celine, and Perkins, Ben, 2015, *Made-to-Order: The Rise of Mass Personalization*, 11th ed., The Deloitte Consumer Review, London, UK, pp. 1–24.
[3] Streitmatter, G. L., and Wiens, G. J., 2019, "Multi-Objective Approach to HRC Manufacturing Environment," ASME Manufacturing Science and Engineering Conference, MSEC2019-2935, Erie, PA, June 10–14 (poster).
[4] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A., 2009, "ROS: An Open-Source Robot Operating System," ICRA Workshop on Open Source Software, **3**(2–3), p. 5.
[5] Chitta, S., Sucan, I., and Cousins, S., 2012, "Moveit!," IEEE Robot. Autom. Mag., **19**(1), pp. 18–19.
[6] Comau e.DO Robot Specifications, 2018, https://edo.cloud/edo-robot/, Accessed October 18, 2019.
[7] Kragic, D., Gustafson, J., Karaoguz, H., Jensfelt, P., and Krug, R., 2018, "Interactive, Collaborative Robots: Challenges and Opportunities," International Joint Conference on Artificial Intelligence, Stockholm, Sweden, July 13–19, pp. 18–25.
[8] Colledanchise, M., and Ogren, P., 2017, "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture and Decision Trees," IEEE Trans. Rob., **33**(2), pp. 372–389.
[9] Roveda, L., Vicentini, F., and Molinari Tosatti, L., 2013, "Deformation-Tracking Impedance Control in Interaction With Uncertain Environments," IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, Nov. 3–7, pp. 1992–1997.
[10] Liu, C., and Tomizuka, M., 2016, "Algorithmic Safety Measures for Intelligent Industrial Co-Robots," IEEE International Conference on Robotics and Automation, Stockholm, Sweden, May 16–21, pp. 3095–3102.
[11] Pedrocchi, N., Vicentini, F., Malosio, M., and Molinari Tosatti, L., 2013, "Safe Human-Robot Cooperation in Industrial Environment," Int. J. Adv. Rob. Syst., **10**(1), pp. 1–13.
[12] Sadrfaridpour, B., and Wang, Y., 2018, "Collaborative Assembly in Hybrid Manufacturing Cells: An Integrated Framework for Human–Robot Interaction," IEEE Trans. Autom. Sci. Eng., **15**(3), pp. 1178–1192.
[13] Pellegrinelli, S., Moro, F. L., Pedrocchi, N., Molinari Tosatti, L., and Tolio, T., 2016, "A Probabilistic Approach to Workspace Sharing for Human-Robot Cooperation in Assembly Tasks," Robotics and Computer-Integrated Manufacturing, CIRP Annals, **65**(1), pp. 57–60.
[14] Ferraguti, F., Talignani Landi, C., Secchi, C., Fantuzzi, C., Nolli, M., and Pesamosca, M., 2017, "Walk-Through Programming for Industrial Applications," Procedia Manufacturing, **11**, pp. 31–38.
[15] Billard, A. G., Calinon, S., and Dillmann, R., 2016, "Learning From Humans," *Springer Handbook of Robotics, Springer Handbooks*, B. Siciliano, and O. Khatib, eds., Springer, Cham, pp. 1995–2014. ISBN: 978-3-319-32550-7.
[16] Neto, P., and Mendes, N., 2013, "Direct Off-Line Robot Programming via a Common CAD Package," Rob. Auton. Syst., **61**(8), pp. 896–910.
[17] Michalos, G., Karagiannis, P., Makris, S., Tokçalar, Ö, and Chryssolouris, G., 2016, "Augmented Reality (AR) Applications for Supporting Human-Robot Interactive Cooperation," Procedia CIRP, **41**, pp. 370–375.